Lecture 19

Tues: HW by 5pm

Weds: Pumpkin drop

Thurs Rieffel ~~[illegible]~~ 7.3
7.5.1 → 7.5.3

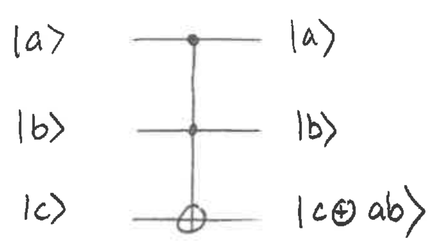
Classical function evaluation

We see that the basic classical function evaluation steps reduce to

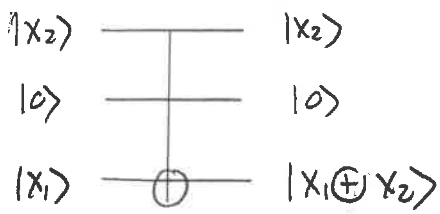
a) addition modulo 2 $(x_1, x_2) \to x_1 \oplus x_2$

b) binary multiplication $(x_1, x_2) \to x_1 x_2$

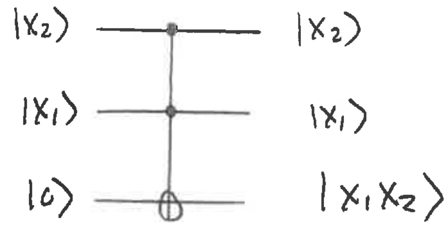
These can both be implemented via Toffoli gates, which in general map computational basis states as:



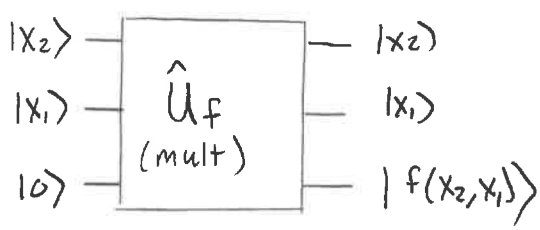
Immediately this gives modular addition



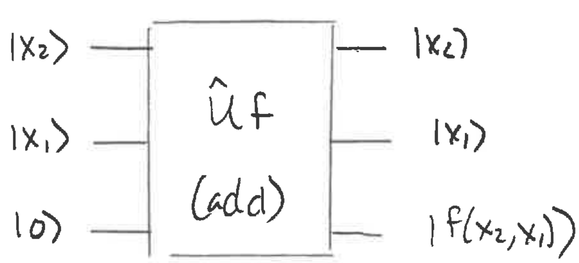
and binary multiplication



The form of the binary multiplication is



where $f(x_2, x_1) = x_2 x_1$ is a binary multiplication function.
With some additional gates modular addition can also be arranged like this:



where $f(x_2, x_1) = x_2 \oplus x_1$

We can try to generalize this to functions that map multiple bits to a single bit. Consider

$$f: n \text{ bits} \to 1 \text{ bit}$$

binary rep $(x_{n-1}, \dots, x_1, x_0) \to f(x_{n-1}, x_{n-2}, \dots, x_1, x_0)$

decimal rep $x \to f(x)$

Examples: 1) f maps $(x_2, x_1, x_0) \to x_0$

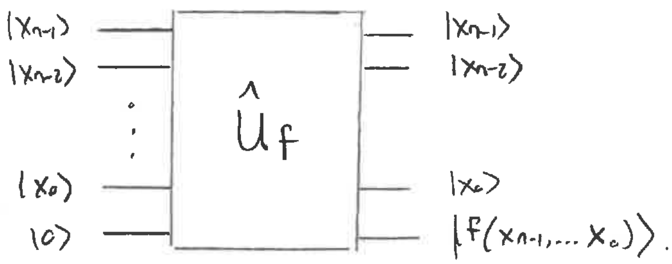
e.g. $0 \to 0$, $1 \to 1$, $2 \to 0$, $3 \to 1$, $4 \to 0$, $5 \to 1$, $\vdots$ } role / checks if input is even or odd

2) f maps $(x_2, x_1, x_0) \to x_2 \oplus x_1 \oplus x_0$

$0 \to 0$, $1 \to 1$, $2 \to 1$, $3 \to 0$, $4 \to 1$, $5 \to 0$, $6 \to 0$, $7 \to 1$ } checks if even or odd number of "1's"

3) f maps $\underbrace{(x_2, x_1, x_0)}_{x} \to \begin{cases} 0 & \text{if 3 divides } x \\ 1 & \text{if 3 does not divide } x. \end{cases}$

Generically we aim to construct an operation which maps



and using $|x\rangle \equiv |x_{n-1}\dots x_0\rangle$ we get that this maps

$$\underbrace{|x\rangle}_{n \text{ qubits}}\underbrace{|0\rangle}_{\text{one qubit}} \xrightarrow{\hat{U}_f} \underbrace{|x\rangle}_{n \text{ qubits}}\underbrace{|f(x)\rangle}_{\text{one qubit.}}$$

Before continuing we need to ensure that this is unitary and to do this there are two steps:

1) describe the effect of $\hat{U}_f$ on states of the form $|x\rangle|1\rangle$

2) rewrite $\hat{U}_f$ as an operator.

Then use the usual rules for checking unitarity.

1 Unitary function evaluation

Consider a function f that maps one bit to a single bit. Define the operator $\hat{U}_f$ via

$$|x\rangle\,|y\rangle \mapsto |x\rangle\,|y \oplus f(x)\rangle$$

where $y = 0, 1$.

a) Verify that the operator

$$\sum_{x'=0}^{1}\sum_{y'=0}^{1} |x'\rangle\,\langle x'| \otimes |y' \oplus f(x')\rangle\,\langle y' \cancel{\oplus f(x')}|$$

maps the input state $|x\rangle\,|y\rangle$ as required.

b) Check that

$$\hat{U}_f = \sum_{x'=0}^{1}\sum_{y'=0}^{1} |x'\rangle\,\langle x'| \otimes |y' \oplus f(x')\rangle\,\langle y' \cancel{\oplus f(x')}|$$

is unitary.

Answer: a) The operator is:

$$\hat{U} = |0\rangle\langle 0| \otimes |0 \oplus f(0)\rangle\langle 0|$$
$$+ |0\rangle\langle 0| \otimes |1 \oplus f(0)\rangle\langle 1|$$
$$+ |1\rangle\langle 1| \otimes |0 \oplus f(1)\rangle\langle 0|$$
$$+ |1\rangle\langle 1| \otimes |1 \oplus f(1)\rangle\langle 1|$$

Consider this acting on $|0\rangle|0\rangle$.

$$\hat{U}|0\rangle|0\rangle = |0\rangle\cancel{\langle 0|0\rangle}^{1} \otimes |0 \oplus f(0)\rangle\cancel{\langle 0|0\rangle}^{1}$$
$$+ |0\rangle\cancel{\langle 0|0\rangle}^{1} \otimes |1 \oplus f(0)\rangle\cancel{\langle 1|0\rangle}^{0}$$
$$+ |1\rangle\cancel{\langle 1|0\rangle}^{0} \otimes \cdots$$
$$+ |1\rangle\cancel{\langle 1|0\rangle}^{0} \otimes \cdots$$
$$= |0\rangle|0 \oplus f(0)\rangle \checkmark$$

Then on $|0\rangle|1\rangle$

$$\hat{U}|0\rangle|1\rangle = |0\rangle\langle 0|0\rangle \otimes |0 \oplus f(0)\rangle\cancel{\langle 0|1\rangle}^{0}$$

$$|0\rangle\langle 0|0\rangle \otimes |1 \oplus f(0)\rangle\cancel{\langle 1|1\rangle}^{1}$$

$$+ |1\rangle\cancel{\langle 1|0\rangle} \otimes \ldots$$

$$= |0\rangle|1 \oplus f(0)\rangle \checkmark$$

continuing gives the same results

b) $$\hat{U}_f^{+} = \sum_{\substack{x'=0 \\ y'=0}}^{1} |x'\rangle\langle x'| \otimes |y'\rangle\langle y' \oplus f(x')|$$

$$\hat{U}_f = \sum_{x'', y''} |x''\rangle\langle x''| \otimes |y'' \oplus f(x'')\rangle\langle y''|$$

So $$\hat{U}_f^{+}\hat{U}_f = \sum_{\substack{x', y' \\ x'', y''}} |x'\rangle\underbrace{\langle x'|x''\rangle}_{\text{only 1 if } x'=x''}\langle x''| \otimes |y'\rangle\langle y' \oplus f(x')|y'' \oplus f(x'')\rangle\langle y''|$$

$$= \sum_{x', y', y''} |x'\rangle\langle x'| \otimes |y'\rangle\langle y' \oplus f(x')|y'' \oplus f(x')\rangle\langle y''|$$

Now $\langle y' \oplus f(x')|y'' \oplus f(x')\rangle$ is only non-zero if $y'=y''$. So the sum is:

$$\sum_{x', y'} |x'\rangle\langle x'| \otimes |y'\rangle\langle y'| = \hat{I} \checkmark$$

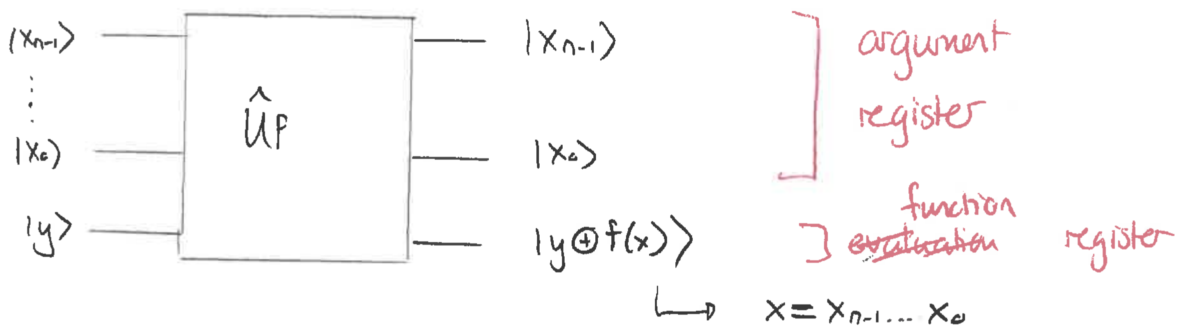
Thus:

> If f maps n bits $\to$ one bit then the operation
>
> $$|x\rangle|y\rangle \xrightarrow{\hat{U}_f} |x\rangle|y\oplus f(x)\rangle$$
>
> is unitary.

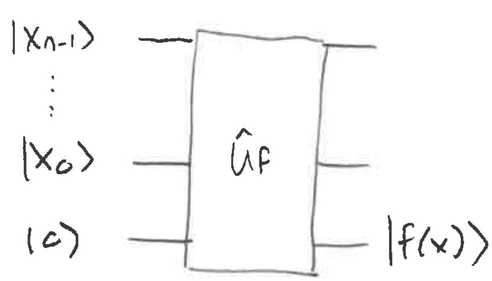
Alternative notations are:

$$\hat{U}_f|x\rangle|y\rangle = |x\rangle|y\oplus f(x)\rangle$$

and the circuit is



This is often called an "oracle" as it will return an output for $f(x)$ if it is provided a relevant input



So, for example one can construct such an oracle to evaluate whether the input is even or odd

Oracle query complexity

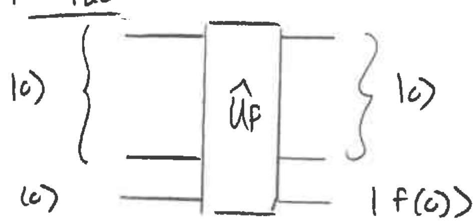
Suppose that one aims to determine a global property of a function, for example whether it is constant or not. One would do this by:

1) supply various inputs to the function

2) evaluate using the oracle on each input

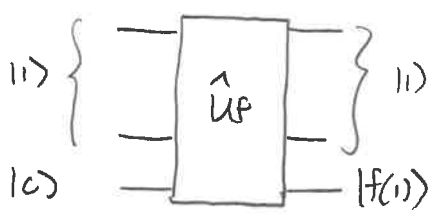
3) measure function register – check whether one always gets same results.

1st Pass



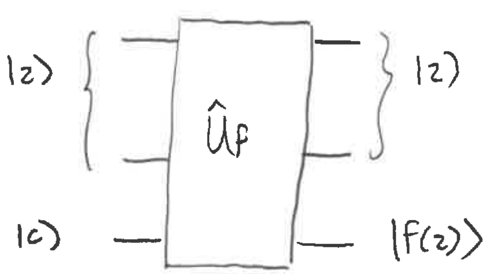
→ measure → gives $f(0)$

2nd Pass



→ measure → gives $f(1)$

3rd pass



→ measure → gives $f(2)$...

So we can repeatedly use the oracle to build up the list

$$\{ f(0), f(1), f(2), \ldots \}$$

We can also count the number of oracle queries needed to perform this task.

To verify with certainty that the function is constant we would in the worst case have to check all 2^n possible inputs. Letting $N = 2^n$ we see that we require N oracle invocations or oracle uses/queries

We describe the difficulty of doing this task in terms of the typical number of oracle invocations required. Here N or 2^n, which grows exponentially in the number of bits on which the function acts. So here

n bit input to function $\longrightarrow$ Requires 2^n oracle queries to determine if function is constant with certainty

This type of growth in the number of oracle queries is exponential and we aim to find algorithms for solving function problems that do not display exponential growth in the number of oracle queries

2 Function evaluation on superpositions

Consider a function that maps n bits to a single bit. Define the operator $\hat{U}_f$ via

$$|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$$

where $y = 0, 1$.

a) Suppose that, rather than, apply the input $|x\rangle = |x_{n-1} \dots x_0\rangle$, to the argument register, one supplies

$$\left[\hat{H} \otimes \cdots \otimes \hat{H}\right] |0 \dots 0\rangle .$$

and function register is in zero.

Determine the state prior to input and the state after the oracle.

b) Suppose that the state of the ~~evaluation~~ function register prior to evaluation is

$$|-\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle\right).$$

Determine the effect of $\hat{U}_f$ on $|x\rangle |-\rangle$ and show that it can be expressed as a multiple of $|x\rangle |-\rangle$.

Answer a)

$$\hat{H}\,|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$$

So

$$\hat{H} \otimes \dots \otimes \hat{H}\ |0\rangle \dots |0\rangle$$

$$= \hat{H}|0\rangle \otimes \dots \otimes \hat{H}|0\rangle$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \dots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$$

$$= \frac{1}{2^{n/2}}\left(|0\rangle + |1\rangle\right) \otimes \dots \otimes \left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

$$\vdots$$

$$= \frac{1}{2^{n/2}}\left[|0 \dots 0\rangle + |0 \dots 01\rangle + |0 \dots 10\rangle + \dots + |1 \dots 10\rangle + |11 \dots 1\rangle\right]$$

$$= \frac{1}{2^{n/2}} \sum_{x=0}^{2^n - 1} |x\rangle$$

Thus prior to input, state is

argument register.

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle\, |0\rangle$$

function register

$$\xrightarrow{U_f} \frac{1}{2^n} \sum_{x=0}^{2^n-1} \hat{U}_f\, |x\rangle |0\rangle$$

$$= \frac{1}{2^n} \sum_{x=0}^{2^n-1} |x\rangle\, |f(x)\rangle$$

Thus after just one oracle query we have evaluated the function on all inputs. But this does not yet give any useful information – a computational basis measurement on the input register will typically only return the function evaluated on one input.

b) $$U_f\, |x\rangle\, |-\rangle = U_f\, |x\rangle \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle\right)$$

$$= U_f \frac{1}{\sqrt{2}} \left(|x\rangle|0\rangle - |x\rangle|1\rangle \right)$$

$$= \frac{1}{\sqrt{2}} \left[\hat{U}_f\, |x\rangle|0\rangle - \hat{U}_f\, |x\rangle\, |1\rangle \right]$$

$$= \frac{1}{\sqrt{2}} \left[|x\rangle\, |f(x)\rangle - |x\rangle\, |1 \oplus f(x)\rangle \right]$$

$$= \frac{1}{\sqrt{2}} |x\rangle \left[|f(x)\rangle - |1 \oplus f(x)\rangle \right]$$

Now if $f(x) = 0$ then the parenthesis gives $|0\rangle - |1\rangle$

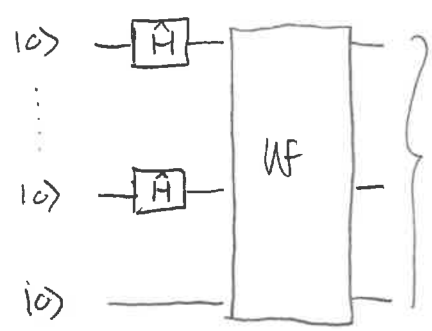
if $f(x) = 1$ " " " " $|1\rangle - |0\rangle = -\left[|0\rangle - |1\rangle\right]$

Thus the parenthesis is $(-1)^{f(x)} \left[|0\rangle - |1\rangle\right]$

So $\quad \hat{U}_f |x\rangle |-\rangle = (-1)^{f(x)} |x\rangle |-\rangle.$ $\blacksquare$

Thus we have two important results that we will combine:

1)



$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$$

2)